Bloom Filtering Cache Misses for Accurate Data Speculation and Prefetching

Jih-Kwon Peir, Shih-Chang Lai, Shih-Lien Lu, Jared Stark, Konrad Lai

peir@cise.ufl.edu

Computer & Information Science and Engineering
University of Florida
It’s the Memory, Stupid

- 3 out of 4 cycles are waiting for memory in Pentium-Pro and Alpha-21164, running TPC-C, - Richard Sites
- Cache Performance: \( f (\text{Hit time}, \text{Miss ratio}, \text{Miss penalty}) \)
- Performance impact due to cache latency:

<table>
<thead>
<tr>
<th>lw  r1 &lt;= 0(r2)</th>
<th>issue</th>
<th>register</th>
<th>addgen</th>
<th>mem1</th>
<th>mem2</th>
<th>hit/miss</th>
<th>commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>add r3 &lt;= r2, r1</td>
<td>stall</td>
<td>stall</td>
<td>stall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Minimum 3-cycle hit latency

Speculative issue for hit
Squashed / re-issued on miss
(Total 3 cycles)
Outline

• Introduction
• Motivation and Related Work
• Bloom Filtering Cache Misses
  – Partitioned-address BF
  – Partial-address BF
• Pipeline Microarchitecture with BF
• Performance Evaluation
• Summary
No data speculation degrades IPC by 15-20% (SPECint2000)
Related Work

- Simple solution: Assume loads always hit L1
- Alpha 21264: 4-bit counter; +1 on hit, -2 on miss
  - Predict hit counter >= 8; mini-recovery if miss
  - Predict miss counter < 8; delay dependent if hit
- Pentium-4: Always hit, “replay” when miss
  - Aggressive way prediction, needs recovery
  - Replay way prediction and load dependent scheduling
- Recovery buffer: Free speculative instructions from scheduling queue
- Hybrid 2-level hit/miss predictor (like branch)
Bloom Filter (BF) - Introduction

• A probabilistic algorithm to test membership in a large set using hashing functions to a bit array.
• A BF quickly filters non-members without querying the large set.
• Filter cache misses
  – Accurate scheduling load dependents
  – Overlap and reduce cache miss penalty
• Filter other table accesses
Partitioned-Address BF

A1 | A2 | A3 | A4
---|----|----|----
R1 | R2 | R3 | R4

**Request Line Addr.**

**Replaced Line Addr.**

- **Cache Miss**
  - **Increment counter on cache miss**
  - **Decrement counter on cache miss**

- **BF1**
- **BF2**
- **BF3**
- **BF4**

**Guarantee miss!**

True if cache miss
False likely cache hit
Partial-Address BF

Requested Line Address

Tag | Index | offset

partial address (p bits)

BF array

Set bit on cache miss

Reset bit on cache miss but no collision

Hit / Miss Detector

Collision Detector

Partial Address (p-bits) of Replaced Cache Line

Collision? (yes/no)

False, miss
True, (may) hit

Guarantee miss!
Virtual-Address BF

- **Benefit of cache miss filtering**
  - Must be *early* before dependent scheduling
  - Must be *accurate*

- **Filter cache miss using virtual address**
  - Must handle address synonym problem
  - Special handling collision detection for physical caches

- **Partial-address (virtual) BFs**
  - Separate collision detection from cache tag path
  - Compare replaced PA with all other PAs with the same page offset
  - Reset BF array only when no match is found
Pipeline Execution with BF

• Virtual address BF filter cache miss 2 cycles earlier, Still one cycle too late for dependent scheduling
  – Delay one cycle for dependent scheduling
  – Always hit, precise recovery for single-cycle speculation
Cache Miss Filtered by BF

- Cache miss filtered by BF, one cycle window
  - Precise recovery, reschedule only dependents (have to wait for miss anyway)
  - No penalty for independent instructions
# Cache Miss Not Filtered

## Load:

<table>
<thead>
<tr>
<th>SCH</th>
<th>REG</th>
<th>AGN</th>
<th>CA1</th>
<th>BF</th>
<th>CA2</th>
<th>H/M</th>
<th>L2 Access</th>
</tr>
</thead>
</table>

### Speculative Window

- **Cache Miss (not filtered)**

## Dependent:

### Dependent:

- SCH REG EXE Flush

## Independent:

### Independent:

- SCH REG EXE WRB CMT
  - 4-cycle penalty

- SCH Flush SCH REG EXE WRB CMT
  - 2-cycle penalty
Prefetching with BF

- BF filtered miss trigger L2 miss 2-cycle earlier
  - L1 miss is guaranteed
  - Applicable to other caches, TLB, branch prediction tables, etc.
## Predictors and Extra Hardware

<table>
<thead>
<tr>
<th>Prediction Method</th>
<th>Additional Table (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioned BF - 3</td>
<td>15360</td>
</tr>
<tr>
<td>Partitioned BF - 4</td>
<td>4480</td>
</tr>
<tr>
<td>Partial BF - 1x</td>
<td>512</td>
</tr>
<tr>
<td>Partial BF - 4x</td>
<td>2048</td>
</tr>
<tr>
<td>Partial BF - 16x</td>
<td>8192</td>
</tr>
<tr>
<td>Partial BF - 64x</td>
<td>32768</td>
</tr>
<tr>
<td>Always-Hit</td>
<td>0</td>
</tr>
<tr>
<td>Counter - 1 (Alpha)</td>
<td>4</td>
</tr>
<tr>
<td>Counter - 128</td>
<td>512</td>
</tr>
<tr>
<td>Counter - 512</td>
<td>2048</td>
</tr>
<tr>
<td>Counter - 2048</td>
<td>8192</td>
</tr>
<tr>
<td>Counter - 8192</td>
<td>32768</td>
</tr>
</tbody>
</table>
Partitioned BFs perform poorly; 97% miss filtered by Partial-16x
Accuracy of Various Predictors

Bloom Filter has NO incorrect-delay, i.e. predict miss always miss
IPC Comparison

![IPC Comparison Chart]

Note: The chart above compares IPC (Instructions Per Cycle) across various applications (Bzip, Gap, Gcc, Gzip, Mcf, Parser, Perl, Twolf, Vortex, Vpr, and Average) for different speculation and cache partitioning techniques. The techniques include No-speculation, Counter-1, Counter-2048, Always-hit, Partition-3, Partial-16x, Partial-16x-DP, Perfect-sch, and Perfect-sch-DP. The chart illustrates the performance impact of these techniques on each application.
Effect of Data Cache Size

![Graph showing IPC improvement for different cache sizes and cache replacement policies.]

- **Perfect-sch-DP**
- **Partial-16x-DP**
- **Partial-16x**
- **Always-hit**

The graph illustrates the IPC improvement (%) for different cache sizes (8KB, 16KB, 32KB, 64KB) and cache replacement policies. The x-axis represents the cache size, and the y-axis represents the IPC improvement percentage.
Impact of RUU to Always-Hit

IPC Improve over Always-Hit (%)

ruu32  ruu64  ruu128

Partial-16x
Partial-16x-DP
Partial-16x-perfect
Partial-16x-DP-perfect
Summary

• Data speculation schedules load dependents without knowing load latency
• Bloom Filter identifies 97% of the misses early using a small 1KB bit array
• 19% IPC improvement over no-speculation, 6% IPC improvement over always-hit method
• Reach 99.7% IPC of a perfect scheduler